STRUTS 2 RESULTS AND RESULT TYPES

As mentioned previously, the **<results>** tag plays the role of a **view** in the Struts2 MVC framework. The action is responsible for executing the business logic. The next step after executing the business logic is to display the view using the **<results>** tag.

Often there is some navigation rules attached with the results. For example, if the action method is to authenticate a user, there are three possible outcomes. (a) Successful Login (b) Unsuccessful Login - Incorrect username or password (c) Account Locked.

In this scenario, the action method will be configured with the three possible outcome strings and three different views to render the outcome. We have already seen this in the previous examples.

But, Struts2 does not tie you up with using JSP as the view technology. Afterall the whole purpose of the MVC paradigm is to keep the layers separate and highly configurable. For example, for a Web2.0 client, you may want to return XML or JSON as the output. In this case, you could create a new result type for XML or JSON and achieve this.

Struts comes with a number of predefined **result types** and whatever we've already seen that was the default result type **dispatcher**, which is used to dispatch to JSP pages. Struts allow you to use other markup languages for the view technology to present the results and popular choices include **Velocity, Freemaker, XSLT** and **Tiles**.

The dispatcher result type:

The **dispatcher** result type is the default type, and is used if no other result type is specified. It's used to forward to a servlet, JSP, HTML page, and so on, on the server. It uses the *RequestDispatcher.forward()* method.

We saw the "shorthand" version in our earlier examples, where we provided a JSP path as the body of the result tag.

```
<result name="success">
   /HelloWorld.jsp
</result>
```

We can also specify the JSP file using a <param name="location"> tag within the <result...> element as follows:

```
<result name="success" type="dispatcher">
   <param name="location">
      /HelloWorld.jsp
   </param >
</result>
```

We can also supply a **parse** parameter, which is true by default. The parse parameter determines whether or not the location parameter will be parsed for OGNL expressions.

The FreeMaker result type:

In this example we are going to see how we can use **FreeMaker** as the view technology. Freemaker is a popular templating engine that is used to generate output using predefined templates. Let us create a Freemaker template file called **hello.fm** with the following contents:

```
Hello World ${name}
```

Here above file is a template where **name** is a paramter which will be passed from outside using the defined action. You will keep this file in your CLASSPATH. Next, let us modify the **struts.xml** to specify the result as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
   <constant name="struts.devMode" value="true" />
   <package name="helloworld" extends="struts-default">

      <action name="hello"

         method="execute">
         <result name="success" type="freemarker">
            <param name="location">/hello.fm</param>
         </result>
      </action>

   </package>

</struts>
```

Let us keep our HelloWorldAction.java, HelloWorldAction.jsp and index.jsp files as we have created them in examples chapter. Now Right click on the project name and click **Export > WAR File** to create a War file. Then deploy this WAR in the Tomcat's webapps directory. Finally, start Tomcat server and try to access URL http://localhost:8080/HelloWorldStruts2/index.jsp. This will give you following screen:

Enter a value "Struts2" and submit the page. You should see the next page

As you can see, this is exactly same as the JSP view except that we are not tied to using JSP as the view technology. We have used Freemaker in this example.

The redirect result type:

The **redirect** result type calls the standard *response.sendRedirect()* method, causing the browser to create a new request to the given location.

We can provide the location either in the body of the <result...> element or as a <param name="location"> element. Redirect also supports the **parse** parameter. Here's an example configured using XML:

```
<action name="hello"

   method="execute">
   <result name="success" type="redirect">
       <param name="location">
         /NewWorld.jsp
      </param >
   </result>
</action>
```

So just modify your struts.xml file to define redirect type as mentioned above and create a new file NewWorld.jpg where you will be redirected whenever hello action will return success. You can check Struts 2 Redirect Action example for better understanding.